On global optimization with indefinite quadratics

Marcia Fampa · Jon Lee · Wendel Melo

21 November 2013

Abstract We present an algorithmic framework for global optimization problems in which the non-convexity is manifested as indefinite quadratic functions. Our solution approach consists of applying a spatial branch-and-bound algorithm, exploiting convexity as much as possible, not only convexity in given convex functions, but also extracted from the indefinite quadratics. A preprocessing stage is proposed to split the indefinite quadratics and rewrite them as a difference of convex quadratic functions, leading to a more efficient spatial branch-and-bound focused on the isolated non-convexity. We investigate several possibilities for splitting quadratics at the preprocessing stage, and prove the equivalence of some of them. Through computational experiments with different categories of test-beds, we analyze how the splitting strategies affect the performance of our algorithm, and find guidelines for choosing amongst them. Numerical comparisons with `Couenne` shows the competitiveness of our approach.

M. Fampa
Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil
E-mail: fampa@cos.ufrj.br

J. Lee
University of Michigan, Ann Arbor, MI, USA
E-mail: jonxlee@umich.edu

W. Melo
Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil
E-mail: wendelmelo@cos.ufrj.br

1 Introduction

We have developed a global-optimization framework aimed at optimization problems (possibly including some integer variables) in which all of the non-convexity is on indefinite quadratic functions. This is a broad framework, within the space of mixed-integer nonlinear programming (MINLP), simultaneously generalizing *convex* MINLP and (indefinite) quadratically-constrained quadratic programming (QCQP). Application areas range across areas such as combinatorial optimization, manufacturing optimization, and sensor-network optimization.

Our approach involves applying a spatial branch-and-bound framework, but taking as much advantage of any convexity as possible — in given convex functions but also in convexity extracted from indefinite quadratics. This is in contrast to approaches that (i) ignore any global convexity (at the peril of accumulating bounds across many low-dimensional convexifications) or that (ii) globally convexify (at the peril of treating any convexity non-optimally).

Our approach for treating indefinite quadratics is by splitting at a preprocessing stage, in a few different natural ways, as a difference of convex (DC) quadratic functions. In this manner, we isolate some inherent non-convexity for focusing our spatial branch-and-bound. There are several different methods for splitting quadratics. We proved the equivalence of some of them, and through computational experiments, we find guidelines for choosing amongst the inequivalent ones.

We have instantiated our methodology as the software `iquad` which can be accessed via the `AMPL` system. As such, `iquad` can receive any nonlinear functions that `AMPL` can receive, and the burden is on the modeler to insure that the nonlinear functions that are not quadratic are convex. Users can choose either `Mosek` or `CSDP` for SDP preprocessing used for some of the splitting strategies. Users can choose any of the convex MIQCP solvers: `Cplex`, `Gurobi` and `Mosek` for solving convex quadratic relaxations (possibly with integer variables) and any of the NLP solvers: `Mosek` and `Ipopt` for solving convex nonlinear relaxations. Moreover, `iquad` supports parallel branch-and-bound, taking advantage of multiple processors in the machine to solve several branch-and-bound subproblems simultaneously.

Our methodology applies to optimization problems of the form

$$\begin{aligned} z := \min\ & f_0(x) + q_0(x)\ , \\ & f_i(x) + q_i(x) \le 0\ , i = 1, 2, \ldots, m\ ; \\ & x \in \mathcal{X}\ , \end{aligned}$$

where the $f_i : \mathbb{R}^n \to \mathbb{R}$ are convex, the $q_i(x) = \frac{1}{2}x'Q_ix$ are pure quadratic, and $\mathcal{X}$ is described in a tractable manner by convex functions and possibly integrality restrictions, but no apparent indefinite quadratics. Note that any affineness in the objective or constraint functions is absorbed by the f_i . If all Q_i are the zero matrix, then we have the case of convex MINLP. If all of the f_i are affine, and $\mathcal{X}$ is all of $\mathbb{R}^n$, we have the case of (indefinite) QCQP.

As for the set $\mathcal{X}$, it might be a mixed-integer linear set:

$$\begin{aligned}&Ax = b\ ,\\ &l_x \le x \le u_x\ ,\\ &x_j \in \mathbb{Z}\ ,\ \text{for } j \in \mathcal{I}\ ,\end{aligned}$$

or a mixed-integer *convex* quadratic set, or a continuous, smooth, convex set, or perhaps just a mixed-integer, smooth, convex set. The only requirement that we really have is that $\mathcal{X}$ be bounded, and optimizing *linear* as well as *convex quadratic* functions on $\mathcal{X}$ (or possibly on a continuous relaxation of $\mathcal{X}$) should be significantly easier than calculating z . In particular, for these four cases, we have MILP solvers (e.g., `Cplex` and `Gurobi`), smooth-NLP solvers (e.g., `Ipopt`), convex MIQCP solvers (e.g., `Cplex`, `Gurobi` and `Mosek`), and convex MINLP solvers (e.g., `Bonmin`). We emphasize that in our approach, we do not restrict our attention to methods that linearly relax convex sets and functions. Rather, we fully exploit convexity by employing appropriate solvers. In such a way, any improvements in the underlying solvers can give us a strong positive impact for `iquad`.

For ease of exposition and to make testing more manageable, we confine our attention to the optimization problem

$$\begin{aligned}z := \min\ & f(x) + q(x)\ ,\\ & x \in \mathcal{X}\ ,\end{aligned} \tag{I}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex, $q(x) := \frac{1}{2}x'Qx$, the symmetric matrix Q is not positive semidefinite, and $\mathcal{X}$ is a bounded subset of $\mathbb{R}^n$. That is, we focus on the case in which the only (quadratic) non-convexity manifests itself as the objective function. But we emphasize that our method is well suited for situations in which there are indefinite quadratic functions present in the constraints as well.

Our first step is to split Q as $Q = P - R$, where P and R are positive *semidefinite*. Our goal is to decompose q into a convex part $p(x) = \frac{1}{2}x'Px$ and a concave part $r(x) = -\frac{1}{2}x'Rx$. We will further treat the concave part by transforming variables, as needed, to make it implicitly separable. Then, we will apply a spatial branch-and-bound (see [35], for example), on this implicit separable concavity, and take full advantage of the convexity extracted from $q(x)$ as $p(x)$. There are several natural ways to do this, and we investigate the possibilities.

Before continuing with the technical development, we will survey some relevant literature. Indefinite quadratic models have broad application in combinatorial optimization. In particular, the Max-Cut problem in edge-weighted graphs is easily modeled this way, and the best-known solution methods exploit such a model (see [32], for example). For example, the state-of-the-art for computing exact ground states of hard Ising spin-glass problems (from statistical physics) relies on such an approach[1]. Moreover, the quadratic assignment problem (which has been used to model many other structured

[1]`http://www.informatik.uni-koeln.de/spinglass/`

combinatorial-optimization problems) is directly and profitably modeled using an indefinite quadratic model (see [15]). On the purely continuous side, concave minimization occurs naturally when we have economies of scale, and so global-optimization techniques have a clear role. In the big space between the purely discrete and purely continuous (i.e., mixed integer nonlinear programming), there are bilinear indefinite quadratic models arising in the full global optimization of cutting stock problems (see [29], for example).

Generally, the approach of spatial branch-and-bound is well known in global optimization (see [1,2], for example), and in mixed integer nonlinear programming (see [18,22,30], for example). The devil is in the details though, and how the general method handles various kinds of functions can vary considerably and with substantial effect. For quadratics in particular, there is considerable literature (see, for example, the surveys [13,22], and the references therein). Of important relevance to our approach are incites gained from [33, 34], where quadratic non-convexity is dynamically identified and combined with disjunctive methodology in a rather sophisticated manner.

In §2, we describe our preprocessing strategy via splitting. In §3, we describe how to apply spatial branch-and-bound to handle the concavity isolated via splitting, by explicitly or implicitly inducing separability. In §4, we describe natural diagonal splitting strategies, whereupon no variable transformation is needed to induce separability. In §5, we describe how the Real Schur Decomposition can be used for splitting. In §6, using semidefinite programming, we describe a family of natural non-diagonal splittings based on minimizing weighted sums of eigenvalues. Although the computational cost would appear to be substantial, we demonstrate that the splitting from the Real Schur Decomposition optimizes *all* splittings of this type. In §7, we describe our computational experiments. In §8, we describe future work.

2 Preprocessing via Splitting

Consider a splitting of Q as $Q = P - R$, where P and R are positive *semidefinite.*

We can calculate the real Schur decomposition of R, namely

$$R = \sum_{i \in N} \lambda_i v_i v_i' \ , \ \text{where } \lambda_i > 0 \text{ for } i \in N \ ,$$

using say `LAPACK`. Now, defining

$$y_i := \sqrt{\lambda_i} v_i' x \ , \ \text{for } i \in N,$$

we reformulate our problem as

$$
\begin{aligned}
z = \min\ & f(x) + \frac{1}{2}x'Px - \frac{1}{2}\sum_{i\in N} y_i^2\ , \\
& x \in \mathcal{X}\ , \\
& y_i = \sqrt{\lambda_i} v_i' x\ ,\ \text{for } i \in N\ , \\
& l_y \le y \le u_y\ ,
\end{aligned}
\tag{$\widetilde{\mathrm{I}}$}
$$

where we calculate the bounds on y by solving the auxiliary programs, for $i \in N$:

$$
\begin{aligned}
l_{y_i} := \sqrt{\lambda_i} \min\ & v_i' x\ , \\
& x \in \mathcal{X}\ ,
\end{aligned}
\tag{L_{y_i}}
$$

and

$$
\begin{aligned}
u_{y_i} := \sqrt{\lambda_i} \max\ & v_i' x\ , \\
& x \in \mathcal{X}\ ,
\end{aligned}
\tag{U_{y_i}}
$$

using an appropriate solver. Note that we really only need valid bounds on the y_i, so we can relax these bounding problems, and simply use a lower bound on l_{y_i} and an upper bound on u_{y_i}. For example, we can partially relax any integrality restrictions (present in $\mathcal{X}$) on x by doing a truncated branch-and-bound search. However, to limit the spatial branch-and-bound search, it is beneficial to have as strong bounds as is practical.

3 Spatial Branch-and-Bound

Our spatial branch-and-bound subproblems will be relaxations of ($\widetilde{\mathrm{I}}$). Every concave term has exactly the same form:

$$
\begin{aligned}
& \omega_i(y_i) := -\frac{1}{2} y_i^2 \\
& l_{y_i} \le y_i \le u_{y_i}\ .
\end{aligned}
$$

So we can give a very explicit formula for the secant under-estimators. We simply replace $-\frac{1}{2}y_i^2$ by a new variable w_i , which we constrain to satisfy the (linear and univariate) secant inequality

$$
-\frac{1}{2}\left((y_i - l_{y_i}) \frac{u_{y_i}^2 - l_{y_i}^2}{u_{y_i} - l_{y_i}} + l_{y_i}^2 \right) \le w_i\ .
$$

So, we have the following relaxation of ($\widetilde{\text{I}}$).

$$\begin{aligned}
&\underline{z} := \min\ c'x + \frac{1}{2}x'Px + \sum_{i\in N} w_i\ , && (\underline{\widetilde{\text{I}}})\\
&x \in \mathcal{X}\ ,\\
&y_i = \sqrt{\lambda_i}v_i'x\ ,\ \text{for } i \in N\ ,\\
&-\frac{1}{2}\left((y_i - l_{y_i})\,\frac{u_{y_i}^2 - l_{y_i}^2}{u_{y_i} - l_{y_i}} + l_{y_i}^2\right) \le w_i\ ,\ \text{for } i \in N\ ,\\
&l_y \le y \le u_y\ .
\end{aligned}$$

If we like, we can rewrite this without the y variables as:

$$\begin{aligned}
&\underline{z} = \min\ c'x + \frac{1}{2}x'Px + \sum_{i\in N} w_i\ , && (\underline{\text{I}})\\
&x \in \mathcal{X}\ ,\\
&-\frac{1}{2}\left(\left(\sqrt{\lambda_i}v_i'x - l_{y_i}\right)\frac{u_{y_i}^2 - l_{y_i}^2}{u_{y_i} - l_{y_i}} + l_{y_i}^2\right) \le w_i\ ,\ \text{for } i \in N\ ,\\
&l_{y_i} \le \sqrt{\lambda_i}v_i'x \le u_{y_i}\ ,\ \text{for } i \in N\ .
\end{aligned}$$

In applying spatial branch-and-bound to (I), using the relaxation ($\underline{\text{I}}$), subproblems will have exactly the same form as ($\underline{\text{I}}$), but with adjusted values of the bounds l_{y_i} and u_{y_i} , for $i \in N$. Once the branching index and branching point are selected, we create two new problems, one where the upper bound u_{y_i} is replaced with ψ_i , and one where the lower bound l_{y_i} is replaced with ψ_i .

Once a branching index is selected, there are many reasonable choices for selecting the branching point ψ_i . For example, if $(\tilde{x}, \tilde{w})$ is the solution to ($\underline{\text{I}}$) (or to a further relaxation of ($\underline{\text{I}}$)), then we can set $\psi_i := \sqrt{\lambda_i}v_i'\tilde{x}$. However, this can be a poor choice if the point $\psi_i := \sqrt{\lambda_i}v_i'\tilde{x}$ is too close to one of the endpoints of the interval $[l_{y_i}\ ,\ u_{y_i}]$. Alternatively, ψ_i can be chosen at the midpoint of the interval $[l_{y_i}\ ,\ u_{y_i}]$. In our earlier computational studies on spatial branch-and-bound, we found it to be effective to use a weighted combination of these two possibilities. That is, setting $\psi_i := \alpha\sqrt{\lambda_i}v_i'\tilde{x} + (1-\alpha)(l_{y_i} + u_{y_i})/2$, with $0 \le \alpha \le 1$ fairly large (e.g., $\alpha = 0.8$).

Regarding choosing the branching index $i \in N$, it can make sense to consider the discrepancy between $-\frac{1}{2}(\sqrt{\lambda_i}v_i'\tilde{x})^2$ and $\tilde{w}_i$. Alternatively, a possible choice of priorities on the $i \in N$ for spatial branch-and-bound could be to choose the i corresponding to the greatest value of $\lambda_i(l_{y_i} - u_{y_i})$.

There is a legitimate concern regarding the density of the secant constraints, which are likely to be fully dense (when written in the x variables). However, for cases in which $|N|$ is small (corresponding to *nearly convex* $q(x)$), this may be quite tolerable. As the degree of non-convexity grows, we expect that the overall dimension n would have to be more modest anyway as we get closer to applying classical spatial branch-and-bound. So this may not be a significant issue.

4 Some Diagonal Splitting Strategies

Splittings for which R is diagonal have the advantage of not needing any reformulation for inducing separability. For diagonal splittings, there is no separability to explicitly or implicitly induce, so we do not carry the burden of dealing with the dense inequalities:

$$l_{y_i} \leq \sqrt{\lambda_i} v_i' x \leq u_{y_i} \ , \text{ for } i \in N \ .$$

There are a few natural diagonal splitting strategies.

Diagonally Dominant: A very simple approach is to let

$$r_i := \max \left\{ 0, \ \ -q_{ii} + \sum_{j:j \neq i} |q_{ij}| \right\} ,$$

for $i = 1, 2, \ldots, n$. Then let $R := \mathrm{Diag}(r_1, r_2, \ldots, r_n)$, so that $P := Q + R$ is diagonally dominant.

Identity: Another possibility, more aggressive in taking convexity into $p(x)$ from $q(x)$ is to let $R := -\min\{0, \lambda_n\}\ I$, where λ_n is the least eigenvalue of Q , in which case $P := Q + R$ is positive semidefinite.

Diagonal SDP: A heavier approach is to let $r := (r_1, r_2, \ldots, r_n)$, take $\mathrm{Diag}(r)$ as a *diagonal* matrix variable, and solve the semidefinite program

$$\begin{aligned} &\min \sum_{i=1}^{n} r_i \ , \\ &P := Q + \mathrm{Diag}(r) \succeq 0 \ , \\ &r \in \mathbb{R}_+^n \ , \end{aligned} \tag{D}$$

minimizing the trace of $R := \mathrm{Diag}(r)$.

This splitting seeks, in some sense, to minimize the separable strict convexity needed to be added to $q(x)$ to render it convex.

5 Splitting via the Real Schur Decomposition

Departing from the realm of diagonal splittings, a very natural approach is to calculate the *Real Schur Decomposition* of Q. That is,

$$Q = \sum_{i=1}^{n} \lambda_i v_i v_i' \ .$$

Splitting this into the parts corresponding to positive and negative eigenvalues, we have $Q = P - R$, where

$$\begin{aligned} P &:= \sum_{i \in P} \lambda_i v_i v_i' \ , \ \text{where } \lambda_i > 0 \text{ for } i \in P \ , \\ R &:= \sum_{i \in N} (-\lambda_i) v_i v_i' \ , \ \text{where } \lambda_i < 0 \text{ for } i \in N \ . \end{aligned}$$

Example 1 Although we are thinking in terms of unstructured matrices Q and determining our splitting computationally, it is instructive to consider a highly structured and common case. If we had

$$Q = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix},$$

with $x' = (x_1', x_2')$, where x_1 and x_2 are in $\mathbb{R}^{n/2}$, then $\frac{1}{2}x'Qx$ is simply the indefinite bilinear function $x_1'x_2$. Then Q has an $n/2$-dimensional eigenspace V_{+1} belonging to its eigenvalue 1 and an $n/2$-dimensional eigenspace V_{-1} belonging to its eigenvalue -1. A basis for V_{+1} is the set of $n/2$ vectors $v_i := (e_i', e_i')'$, $i = 1, \ldots, n/2$, and a basis for V_{-1} is the set of $n/2$ vectors $v_{\frac{n}{2}+i} := (e_i', -e_i')'$, $i = 1, \ldots, n/2$. Then we have $Q = P - R$, with

$$\begin{aligned} P &:= \sum_{i=1}^{\frac{n}{2}} v_i v_i' = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}, \\ R &:= \sum_{i=\frac{n}{2}+1}^{n} v_i v_i' = \begin{pmatrix} I & -I \\ -I & I \end{pmatrix}. \end{aligned}$$

If we choose to explicitly reformulate the concave part separably, then corresponding to the eigenvectors $v_{\frac{n}{2}+i} := (e_i', -e_i')'$, $i = 1, \ldots, n/2$, we define

$$y_i := v_{\frac{n}{2}+i}' x = x_i - x_{\frac{n}{2}+i} = x_{1,i} - x_{2,i} \ , \ \text{for } i = 1, \ldots, n/2.$$

This is precisely the well-known method of "inducing separability" of products as a prelude to performing a piecewise-linear approximation, which can be found for example in [11, p. 579].

6 Minimizing weighted sums of eigenvalues

There is another very interesting and natural class of splittings with non-diagonal R. We consider splittings $Q = P - R$ that solve

$$\begin{aligned} \min \ & \sum_{i=1}^{k} m_i \lambda_i(R) \ , && \text{(WESDP)} \\ & P := Q + R \succeq 0 \ ; \\ & R \succeq 0 \ , \end{aligned}$$

where $m_1 \geq m_2 \geq \cdots m_k > 0 =: m_{k+1}$.

Of course if we set $k := n$ and $m_i := 1$, for $i = 1, 2, \ldots, n$, we are minimizing the trace of R . Because we allow non-diagonal R, as compared to the 'Diagonal SDP' option of §4, we should expect to find better splittings, but at an apparently higher computational cost.

Alternatively, if we set $k := 1$ and $m_1 := 1$, we are simply minimizing the maximum eigenvalue of R , whereupon (WESDP) can be recast as

$$\begin{aligned} \min \ & z \ , \\ & zI - R \succeq 0 \ ; \\ & P := Q + R \succeq 0 \ ; \\ & R \succeq 0 \ . \end{aligned}$$

But we can apparently do better than this by choosing $m_1 >> m_2 >> \cdots >> m_n > 0$. In such a case, we are finding an R having a lexically minimum list of eigenvalues.

In this section, we demonstrate that *all* of these weighted-eigenvalue minimization problems (WESDP) are actually solved, and very efficiently, by using the splitting determined by the Real Schur Decomposition of §5. So in fact, the splitting determined by the Real Schur Decomposition is fundamental.

Theorem 1 *For any given parameters m_i, $i = 1, \ldots, k$, such that $m_1 \geq m_2 \geq \cdots m_k > 0 =: m_{k+1}$, the splitting determined by the Real Schur Decomposition solves the weighted-eigenvalue minimization problem (WESDP).*

Lemma 1 *(WESDP) is equivalent to the SDP:*

$$\begin{aligned} \min \ & \sum_{i=1}^{k} i z_i + \sum_{i=1}^{k} Tr(V_i) \ , && \text{(PSDP)} \\ & z_i I + V_i - (m_i - m_{i+1}) R \succeq 0 \ , \quad i = 1, 2, \ldots, k \ ; \\ & Q + R \succeq 0 \ ; \\ & R \succeq 0 \ ; \\ & V_i \succeq 0 \ , \quad i = 1, 2, \ldots, k \ . \end{aligned}$$

and its dual can be written as

$$\begin{aligned}
&\max\ Q \bullet X\ , && \text{(DSDP)}\\
&Tr(Y_i) = i\ ,\quad i = 1,2,\ldots,k\ ;\\
&\sum_{i=1}^{k}(m_i - m_{i+1})Y_i + X \succeq 0\ ;\\
&0 \preceq Y_i \preceq I\ ,\quad i = 1,2,\ldots,k\ ;\\
&X \preceq 0\ .
\end{aligned}$$

Proof Applying a description in [3], (WESDP) is a convex minimization problem and can be re-formulated as (PSDP), or, equivalently

$$\begin{aligned}
&\min\ \sum_{i=1}^{k} i(\bar{z}_i - \underline{z}_i) + \sum_{i=1}^{k}\mathrm{Tr}(V_i)\ , && \text{(PSDP')}\\
&(\bar{z}_i - \underline{z}_i)I + V_i - (m_i - m_{i+1})R - W_i = 0\ ,\quad i = 1,2,\ldots,k\ ;\\
&Q + R - S = 0\ ;\\
&R, S \succeq 0\ ;\\
&V_i, W_i \succeq 0\ ,\quad i = 1,2,\ldots,k\ ;\\
&\bar{z}_i, \underline{z}_i \geq 0\ ,\quad i = 1,2,\ldots,k\ .
\end{aligned}$$

It is convenient to write (PSDP') as a standard-form SDP

$$\begin{aligned}
&\min\ F_0 \bullet \Gamma\ , && \text{(SPSDP)}\\
&F_j \bullet \Gamma = c_j\ ,\quad j = 1,2,\ldots,m\ ;\\
&\Gamma \succeq 0\ ,
\end{aligned}$$

where " $\bullet$ " is the usual inner product of matrices:

$$A \bullet B := \sum_{i,j} A_{ij}B_{ij} = \mathrm{Tr}(A'B).$$

Define

$$\Gamma := \mathrm{Diag}\left(\bar{Z}, \underline{Z}, S, R, W_1, W_2, \ldots, W_k, V_1, V_2, \ldots, V_k\right),$$

where $\bar{Z}$ is a diagonal matrix with $\bar{Z}_{ii} = \bar{z}_i$, for $i = 1,\ldots,k$ (analogous definition applies for $\underline{Z}$), and

$$F_0 := \mathrm{Diag}\left(G, -G, 0, 0, 0, 0, \ldots, 0, I, I, \ldots, I\right),$$

where G is a diagonal matrix, with $G_{jj} = j$, for $j = 1,\ldots,k$. The objective function of problem (PSDP') can then be written as $F_0 \bullet \Gamma$.

We now define $k+1$ groups of $n(n+1)/2$ matrices. Each group is used to formulate an equality constraint of problem (PSDP'). The first n matrices of each group are used to determine the diagonal values of the matrices on the left hand side of the constraints and the other $n(n-1)/2$ matrices of the group are used to determine the non-diagonal values. The notation E_i is used in the following to represent the diagonal matrix with element (i,i) equal to one and all the others equal to zero. Furthermore, $E_{\gamma\beta}$ denotes the symmetric matrix with elements (γ,β) and (β,γ) equal to one and all the others equal to zero.

For the first constraint in (PSDP'),

$$(\bar{z}_1 - \underline{z}_1)I + V_1 - (m_1 - m_2)R - W_1 = 0 \ ,$$

we define

$$F_j := \text{Diag}\,(E_1, -E_1, 0, -\tilde{m}_1 E_j, -E_j, 0, \ldots, 0, E_j, 0, \ldots, 0)\,,$$

for $j = 1, \ldots, n$, and

$$F_{n+j} := \frac{1}{2}\text{Diag}\,(0, 0, 0, -\tilde{m}_1 E_{\gamma\beta}, -E_{\gamma\beta}, 0, \ldots, 0, E_{\gamma\beta}, 0, \ldots, 0)\,,$$

for $j = 1, \ldots, n(n-1)/2$, where $\tilde{m}_1 := m_1 - m_2$. The indexes γ, β satisfy $\gamma, \beta \in \{1, \ldots, n\}$, with $\gamma > \beta$. Each pair (γ, β) uniquely corresponds to an index j.

The constraint $(\bar{z}_1 - \underline{z}_1)I + V_1 - (m_1 - m_2)R - W_1 = 0$ can then be written as $F_j \bullet \Gamma = 0$, for $j = 1, \ldots, n(n+1)/2$.

An analogous group of matrices are used to formulate each of the k first equality constraints of (PSDP'). For the last constraint,

$$(\bar{z}_k - \underline{z}_k)I + V_k - (m_k - m_{k+1})R - W_k = 0 \ ,$$

we define

$$F_{\nu_1+j} := \text{Diag}\,(E_k, -E_k, 0, -\tilde{m}_k E_j, 0, 0, \ldots, -E_j, 0, 0, \ldots, E_j)\,,$$

for $j = 1, \ldots, n$, and

$$F_{\nu_1+n+j} := \frac{1}{2}\text{Diag}\,(0, 0, 0, -\tilde{m}_k E_{\gamma\beta}, 0, 0, \ldots, -E_{\gamma\beta}, 0, 0, \ldots, E_{\gamma\beta})\,,$$

for $j = 1, \ldots, n(n-1)/2$, where $\tilde{m}_k := m_k - m_{k+1}$ and ν_1 is the number of matrices used to formulate the $k-1$ first constraints.

Finally, to formulate the constraint $Q + R - S = 0$, we define

$$F_{\nu_2+j} := \text{Diag}\,(0, 0, E_j, -E_j, 0, 0, \ldots, 0, 0, 0, \ldots, 0)$$

and $c_{\nu_2+j} := Q_{jj}$, for $j = 1, \ldots, n$, and

$$F_{\nu_2+n+j} := \frac{1}{2}\text{Diag}\,(0, 0, E_{\gamma\beta}, -E_{\gamma\beta}, 0, 0, \ldots, 0, 0, 0, \ldots, 0)$$

and $c_{\nu_2+n+j} := Q_{\gamma\beta}$, for $j = 1, \ldots, n(n-1)/2$, where ν_2 is the number of matrices used to formulate the k first constraints.

The dual of (SPSDP) is

$$\begin{aligned} &\max\ c'\delta\ , \\ &F(\delta) \succeq 0\ , \end{aligned} \tag{SDSDP}$$

where

$$F(\delta) := F_0 - \sum_{i=1}^{m} \delta_i F_i\ .$$

Considering the definitions above for F_0 and F_j, $j = 1, \ldots, m$, where $m = (k+1) \times (n(n+1)/2)$, we have:

$$F(\delta) = \text{Diag}\left(U, -U, -X, \tilde{X}, Y_1, Y_2, \ldots, Y_k, I - Y_1, I - Y_2, \ldots, I - Y_k\right),$$

where U is a diagonal matrix with $U_{ii} = i - \text{Tr}(Y_i)$, for $i = 1, \ldots, k$, and $\tilde{X} := \sum_{i=1}^{k}(m_i - m_{i+1})Y_i + X$.

The dual problem (SDSDP) can then be written in the equivalent form (DSDP). □

Lemma 2 *Let R^*, V_i^*, and z_i^*, for $i = 1, 2, \ldots, k$, be a feasible solution to the primal problem (PSDP). Let X^* and Y_i^*, for $i = 1, \ldots, k$, be a feasible solution to the dual problem (DSDP). Then the solutions are primal and dual optimal, if and only if they satisfy the complementarity constraints*

$$\left(\sum_{i=1}^{k}(m_i - m_{i+1})Y_i^* + X^*\right) R^* = 0\ ; \tag{1}$$

$$(I - Y_i^*)V_i^* = 0\ , \quad i = 1, 2, \ldots, k\ ; \tag{2}$$

$$\left(z_i^* I + V_i^* - (m_i - m_{i+1})R^*\right) Y_i^* = 0\ , \quad i = 1, 2, \ldots, k\ ; \tag{3}$$

$$(Q + R^*)X^* = 0\ . \tag{4}$$

Proof The complementarity conditions for the standard primal and dual pair (SPSDP) and (SDSDP) are given by $F(\delta)\Gamma = 0$. Using the expressions for $F(\delta)$ and Γ from the proof of the previous lemma, we obtain the optimality conditions for problems (PSDP) and (DSDP). □

Lemma 3 *Let*

$$\sum_{i=1}^{n} \lambda_i v_i v_i' = \sum_{i \in P} \lambda_i v_i v_i' - \sum_{i \in N} (-\lambda_i) v_i v_i'\ ,$$

be the Real Schur Decomposition of Q , where $\lambda_i > 0$ for $i \in P$ and $\lambda_i < 0$ for $i \in N$. So $Q = \tilde{P} - \tilde{R}$, where $\tilde{P} := \sum_{i \in P} \lambda_i v_i v_i'$ and $\tilde{R} := \sum_{i \in N}(-\lambda_i) v_i v_i'$.

Without loss of generality, let $N = \{1, \ldots, \bar{n}\}$ and $(-\lambda_1) \geq (-\lambda_2) \geq \ldots \geq (-\lambda_{\bar{n}})$.

Let

$$\begin{aligned}
&\bar{R} = \tilde{R}\ ;\\
&\bar{z}_i = (m_i - m_{i+1})(-\lambda_{i+1})\ , \quad i = 1,2,\ldots,\bar{n}-1\ ;\\
&\bar{z}_i = 0\ , \quad i = \bar{n}, \bar{n}+1,\ldots,k\ ;\\
&\bar{V}_i = \sum_{j=1}^{\underline{i}} \left((m_i - m_{i+1})(-\lambda_j) - \bar{z}_i\right) v_j v_j'\ , \quad i = 1,2,\ldots,k\ ;
\end{aligned}$$

where $\underline{i} = \min\{i, \bar{n}\}$, *and*

$$\begin{aligned}
\bar{Y}_i &= \sum_{j=1}^{i} v_j v_j'\ , \quad i = 1,2,\ldots,k\ ;\\
\bar{X} &= -\sum_{i=1}^{k} (m_i - m_{i+1})\bar{Y}_i\ .
\end{aligned}$$

Then $\bar{R}$, $\bar{V}_i$, *and* $\bar{z}_i$, *for* $i = 1,2,\ldots,k$, *is an optimal solution for (PSDP) and,* $\bar{X}$ *and* $\bar{Y}_i$, *for* $i = 1,\ldots,k$, *is an optimal solution for (DSDP).*

Proof Let us first verify that the solution given by $\bar{R}$, $\bar{V}_i$, and $\bar{z}_i$, for $i = 1,2,\ldots,k$, is feasible to problem (PSDP).

Clearly $\bar{R} \succeq 0$. The positive semidefinitiness of the matrices $\bar{V}_i$ results from the ordering of the eigenvalues of $\bar{R}$, $(-\lambda_1) \geq (-\lambda_2) \geq \ldots \geq (-\lambda_{\bar{n}}) > 0$. Since $Q + \bar{R} = \tilde{P}$, we also have $Q + \bar{R} \succeq 0$.

Concerning the constraints $z_i I + V_i - (m_i - m_{i+1})R \succeq 0\ , \quad i = 1,2,\ldots,k$, we divide the analysis into two cases. For $i < \bar{n}$, we have

$$\begin{aligned}
&\bar{z}_i I + \bar{V}_i - (m_i - m_{i+1})\bar{R}\\
&\quad = (m_i - m_{i+1})(-\lambda_{i+1})I + \textstyle\sum_{j=1}^{i} \left((m_i - m_{i+1})(-\lambda_j) - \bar{z}_i\right) v_j v_j'\\
&\qquad -(m_i - m_{i+1}) \textstyle\sum_{j=1}^{\bar{n}} (-\lambda_j) v_j v_j\\
&\quad = (m_i - m_{i+1})(-\lambda_{i+1})I - \textstyle\sum_{j=1}^{i} \bar{z}_i v_j v_j' - (m_i - m_{i+1}) \textstyle\sum_{j=i+1}^{\bar{n}} (-\lambda_j) v_j v_j\\
&\quad = (m_i - m_{i+1}) \left((-\lambda_{i+1})I - \textstyle\sum_{j=1}^{i} (-\lambda_{i+1}) v_j v_j' - \textstyle\sum_{j=i+1}^{\bar{n}} (-\lambda_j) v_j v_j\right).
\end{aligned} \tag{5}$$

Therefore, the positive semidefinitiness of $\bar{z}_i I + \bar{V}_i - (m_i - m_{i+1})\bar{R}$ also results from the ordering of the eigenvalues of $\bar{R}$.

When $i \geq \bar{n}$, $\bar{z}_i = 0$ and $\bar{V}_i = (m_i - m_{i+1})\bar{R}$. The constraints are then clearly satisfied.

It is straightforward to verify that the solution given by $\bar{X}$ and $\bar{Y}_i$, for $i = 1,\ldots,k$, is feasible to (DSDP).

We finally verify that the solutions satisfy the complementarity constraints. Constraint (1) is satisfied from the definition of $\bar{X}$. Constraints (2) are satisfied because the orthonormality of the eigenvectors of $\bar{R}$ results in $\bar{V}_i = \bar{Y}_i \bar{V}_i$.

Concerning the constraints (3), we again divide the analysis into two cases. For $i < \bar{n}$, using the last expression in (5), we have

$$\begin{aligned}(\bar{z}_i I + \bar{V}_i - (m_i - m_{i+1})\bar{R})\bar{Y}_i &= (m_i - m_{i+1})\left((-\lambda_{i+1})I\right.\\ &\quad - \textstyle\sum_{j=1}^{i}(-\lambda_{i+1})v_j v_j'\\ &\quad \left. - \textstyle\sum_{j=i+1}^{\bar{n}}(-\lambda_j)v_j v_j\right)\left(\textstyle\sum_{j=1}^{i} v_j v_j\right)\\ &= 0.\end{aligned}$$

The last equality also results from the orthonormality of the eigenvectors of $\bar{R}$.

When $i \geq \bar{n}$, $\bar{z}_i = 0$ and $\bar{V}_i = (m_i - m_{i+1})\bar{R}$. Constraints (3) are then clearly satisfied.

The last constraint (4) is satisfied because the eigenvectors of $\tilde{P} = Q + \bar{R}$ are orthogonal to the eigenvectors of $\bar{X}$. □

Theorem 1 now follows.

7 Computational Experiments

We coded our software `iquad` in `C++`. In our numerical experiments, we used `Mosek` for solving the convex QP relaxations at the nodes of the branch-and-bound enumeration tree, and also for solving the SDP program (D), for the diagonal SDP splitting strategy described in §4. For calculating eigenvalues and real Schur decompositions, we used `LAPACK` and `BLAS` routines from the Intel Math Kernel Library (`Intel MKL`).

We present computational results for four problem categories. The tests were executed on the Flux computing cluster at the University of Michigan. Flux is a Linux-based HPC cluster based on the Intel platform, consisting of Intel Xeon processors, operating at 2.6 GHz, each of which can access up to 48GB of RAM (though mostly we confined ourselves to 4GB). Each run was executed on a single processor, using a time limit of 2 hours per instance. The absolute and relative convergence tolerances used for all experiments were 10^{-4} and 10^{-3}, respectively. It is important to note that the time of preprocessing is not considered in the time limit of 2 hours, because our main goal in these experiments is to analyze how the different splitting methods affect the solvability of the problems. Nevertheless, we note that the time to preprocess the problems, i.e., to compute the positive semidefinite matrix R, such that $Q = P - R$, is insignificant for all splitting strategies ($\ll 1$ second), except for Diagonal SDP, where we solve the SDP program (D). More comments about the preprocessing time for Diagonal SDP will be made later in this section.

The aim of our experiments is to compare the four different splitting methods proposed in §4 and §5 as preprocessing at the root problem of a spatial branch-and-bound search, namely:

- Diagonally Dominant ("D-Dom").
- Identity ("Identity").
- Diagonal SDP ("D-SDP").

– Real Schur Decomposition (“RSD”).

We also compare our methods to the well-known generic spatial branch-and-bound code `Couenne` [8,17]. The convergence tolerances for `Couenne` were set equal to the ones used for `iquad`. We also set `Cplex` as `Couenne`’s LP solver.

The four categories of the test problems are described as:

– “BoxQP” problems. These are 99 randomly generated box-constrained quadratic programs with Q of various density and n varying from 20 to 125. They are challenging benchmark problems commonly used to compare methods in the literature (see [14]).
– “R-BiqMac” problems. These are 343 problems from the Biq Mac (Binary quadratic and Max-cut) Library[2], where the integrality constraints are relaxed, i.e. $x_i \in \{0,1\}$ is replaced by $x_i \in [0,1]$. These problems constitute our second set of box-constrained quadratic programs, with n varying from 30 to 500.
– “GLOBALLib” problems. These are problems from the repository of global optimization instances GLOBALLib[3]. Of the 413 problems from GLOBALLib, we selected the 83 problems with non-convex quadratic objective function and linear constraints. The number of variables on these problems varies from 2 to 79.
– “Random” problems. These are randomly generated problems with non-convex quadratic objective function and linear constraints. We used the technique proposed by Calamai, Vicente and Júdice (see [16]) to generate instances of five sizes ($n = 20, 40, 60, 80, 100$) and belonging to the three following groups:
  – Problems where the objective function is a sum of a bilinear and a convex quadratic function of disjoint subsets of $n/2$ variables.
  – Problems where the objective function is a sum of a concave and a convex quadratic function of disjoint subsets of $n/2$ variables.
  – Problems where the objective function is a sum of a bilinear, a concave quadratic and a convex quadratic function of disjoint subsets of about $n/3$ variables.

  The number of linear constraints in the problems is equal to $1.5n$. We generated four instances of each size and each group totalizing 60 random instances.

Table 1 presents comparisons among `Couenne` and the four splitting strategies of `iquad`. The percentage of problems in each category that were solved to optimality within different time limits are reported.

From the results presented in Table 1, we conclude that:

– `iquad` does not succeed on the relaxed BiqMac problems R-BiqMac, for any splitting strategy. For these instances, `Couenne` is more successful, although it only solves less than 50% of the problems in the time limit of 2 hours.

[2]`http://biqmac.uni-klu.ac.at/biqmaclib.html`
[3]`http://www.gamsworld.org/global/globallib.htm`

		Splitting Strategy				
Test-Bed	Time(m)	RSD	D-SDP	D-Dom	Identity	`Couenne`
R-BiqMac	30	0.29	8.45	4.37	2.62	47.52
	60	0.58	10.79	4.37	2.92	48.40
	90	0.87	11.95	4.37	3.50	48.40
	120	0.87	12.24	4.66	3.50	48.69
BoxQP	30	13.13	61.62	9.09	50.51	30.30
	60	14.14	65.66	11.11	51.52	32.32
	90	17.17	65.66	12.12	52.53	33.33
	120	17.17	68.69	14.14	52.53	33.33
GLOBALLib	30	100.00	100.00	97.59	100.00	75.90
	60	100.00	100.00	97.59	100.00	75.90
	90	100.00	100.00	97.59	100.00	75.90
	120	100.00	100.00	97.59	100.00	75.90
Random	30	76.67	25.00	15.00	3.33	48.33
	60	78.33	25.00	18.33	5.00	48.33
	90	83.33	25.00	26.67	5.00	51.67
	120	83.33	25.00	26.67	5.00	51.67

Table 1 Percentage of problems solved (%)

- `iquad` with the D-SDP splitting strategy is the best method on BoxQP problems. Also, for these problems, the times for solving the SDP programs at preprocessing are not big. The geometric mean of the preprocessing times is about 3 seconds, corresponding to less than 4% of the geometric mean of `iquad`'s running times. Nevertheless, we should mention that the Identity splitting strategy, which is cheap to compute, may also be a good alternative for these instances: Identity is better than `Couenne` and not so much worse than D-SDP.
- `iquad` is very good on GLOBALLib problems, for all splitting strategies, always showing better performance than `Couenne`.
- `iquad` with the Real Schur Decomposition splitting strategy, RSD, is the best method on the Random problems, significantly better than all the others.
- Overall the problem categories, D-SDP almost always dominates both D-Dom and Identity. Therefore, if we could find a cheap way to preprocess for the D-SDP splitting strategy, then we could discard D-Dom and Identity. However, the solution of the SDP program (D) may be expensive. It is in fact an impediment for many R-BiqMac instances, leading Identity or D-Dom to be the best alternative in some cases.
- RSD performs very well on instances with linear constraints, being the best alternative on both sets of test problems in this category: GLOBALLib and Random. On the other hand, RSD does not present good results for the instances with only box constraints, R-BiqMac and BoxQP. In this case, the diagonal splitting strategies show, in general, better results.

In Table 2, we compare the best lower bounds computed by `iquad` every 30 minutes up to 2 hours, when each one of the splitting methods is used. We report average normalized gaps for each category of test problems. The

		Splitting Strategy			
Test-Bed	Time(m)	RSD	D-SDP	D-Dom	Identity
R-BiqMac	30	99.50	0.70	19.22	7.13
	60	99.16	0.69	19.86	7.20
	90	98.84	0.66	20.16	7.22
	120	98.81	0.65	20.54	7.25
BoxQP	30	81.53	0.19	41.03	0.69
	60	78.98	0.16	40.91	0.63
	90	77.98	0.15	40.48	0.60
	120	77.44	0.14	39.87	0.58
GLOBALLib	30	0.00	0.00	2.41	0.00
	60	0.00	0.00	2.41	0.00
	90	0.00	0.00	2.41	0.00
	120	0.00	0.00	2.41	0.00
Random	30	0.03	8.34	26.10	93.33
	60	0.03	8.03	25.06	91.67
	90	0.02	7.86	23.44	91.67
	120	0.02	7.72	23.05	91.67

Table 2 Normalized gap (%)

normalized gap is defined as the percentage of the worst gap computed, considering the four splitting methods. For example, for the splitting method s, the normalized gap is given by $\frac{opt-lb_s}{opt-lb_{min}} \times 100$, where lb_s is the lower bound computed by `iquad` when using s, lb_{min} is the worst lower bound given by all splitting methods, and *opt* is the optimal solution value of the problem (or the best known solution value). The average results reported in Table 2 take into account only instances for which the worst gap amongst the four splitting strategies is nonzero.

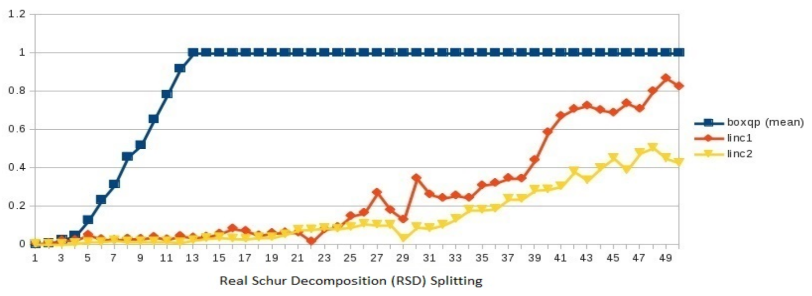
Our conclusions about the results presented in Table 2 are:

– Although `iquad` does not perform well on R-BiqMac problems for any splitting strategy when compared to `Couenne`, we still can observe that among the four strategies, D-SDP is the most promising one, when considering the gap for unsolved problems, and also when considering the percentage of problems solved, reported in Table 1.
– For BoxQP problems, D-SDP leads to the best method, also from both points of view (percentage of solved problems and gap for unsolved problems). However, Identity also presents very good results and is currently more practical in some cases.
– For GLOBALLib problems, `iquad` significantly outperforms `Couenne`, for all of our splitting strategies. However, we can rule out D-Dom as a good method for these problems, because of the big gaps left for the small number of unsolved problems.
– For Random problems, RSD is a strong winner for problems solved, even over `Couenne`, and it also leaves small gaps for the unsolved ones.

Our observations about the performance of the methods on the instances from our test-beds, may be used as guidelines for choosing one of the proposed splitting strategies. We note that, in general, our non-diagonal splitting strategy RSD is a good method for the problems with linear constraints (GLOBALLib and Random), and when RSD is not a good alternative (e.g., for BoxQP), D-SDP is good, though it may be too expensive. For BoxQP problems, Identity is a good substitute for D-SDP. For BiqMac, even though D-SDP is the best splitting strategy for `iquad`, it is still far worse than `Couenne`.

Finally, we made a further set of experiments aimed at understanding how our methods respond across varying amounts of quadratic concavity. We generated random instances of varying degrees of concavity on the objective function, which is measured by the number of negative eigenvalues of Q. First, we generated four basic random instances with $n = 50$, two of them with only box constraints ("boxqp$_1$" and "boxqp$_2$") and the two others with linear constraints ("linc$_1$" and "linc$_2$"). Next, we changed the sign of the eigenvalues of the matrices Q (preserving the eigenvectors), generating 4×50 matrices with varying number of negative eigenvalues from 1 to 50. Each matrix defines an instance of our test-bed. We solved the instances with `iquad`, using all splitting strategies. In Figure 1, we report CPU times for both RSD, the non-diagonal splitting, and D-SDP (the diagonal splitting with best performance on these instances). The horizontal axis of the graphs indicates the number of negative eigenvalues of Q, and the vertical axis indicates the fraction of the maximum CPU time (2 hours) to solve the instance, computed in a logarithmic scale. We note that the results obtained for both box-constrained instances were very similar, so we chose to present only one line for these instances indicating their average CPU times.

An interesting observation about the results presented in Figure 1 is that for our non-diagonal method (RSD), we see a strong dependence on the number of negative eigenvalues, matching the intuition that the number of concave directions for branching is equal to this number. For the diagonal methods, we have n concave directions, regardless of the number of negative eigenvalues, so the performance of the algorithm does not worsen when the number of negative eigenvalues increases. For small number of negative eigenvalues (8 or less), RSD is always the best strategy, for all instances considered. When the number of negative eigenvalues increases, the best strategy is dependent on the problem category. Comparing the two categories, we can clearly observe again the better performance of the non-diagonal method RSD when the problem has linear constraints. In this case, even with the increase of the computational effort with the number of negative eigenvalues, RSD always significantly outperforms all the diagonal splittings. For the box-constrained problems, D-SDP is the best strategy, except for the problems with a small number of negative eigenvalues. Again, we observed with the experiments that Identity's performance is almost as good as D-SDP's on these instances. For the problems with a very small number of negative eigenvalues, the diagonal methods have a bad performance, even on the box-constrained problems, especially Identity. The relaxations given by Identity are too weak in this case.



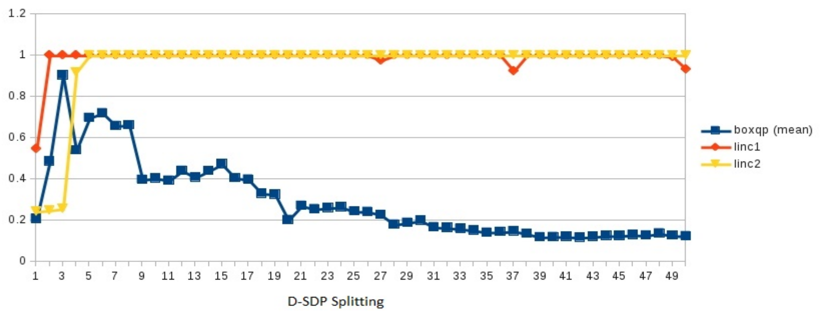


Fig. 1 Effect of the degree of concavity in iquad

8 Future Enhancements

At a high level, the most important features to add are efficient exploitation of integer variables and the application of our methodology to indefinite quadratics in the constraints.

We note that during the spatial branch-and-bound, we may strengthen the model of $\mathcal{X}$ for a subproblem, in particular if there are integer variables. In such cases, it may be valuable to resolve L_{y_i} and U_{y_i} to compute improved bounds for the relaxation ($\underline{\text{I}}$).

Furthermore, as the spatial branch-and-bound proceeds, whenever an x_j variable gets fixed at a subproblem of the branch-and-bound search (which can be quite likely for integer variables), it may be beneficial to compute a new splitting for the reduced quadratic form, though we have not tested this.

If there are no integer variables (i.e., $\mathcal{I} = \emptyset$) and the constraints describing $\mathcal{X}$ are linear, an indefinite QP solver can be used to generate solutions with correct objective value (that is *upper bounds*). This could be of significant help in a spatial branch-and-bound.

In [33,34], different methods were developed for handling quadratics. Quadratics were thought of in the form $q(x,Y) = c'x + \frac{1}{2}\langle Q, Y\rangle$, where $Y := xx'$.

So, in the higher dimensional space of both $x \in \mathbb{R}^n$ and $Y \in \mathbb{R}^{n\times n}$, q is linear, and the quadratic non-convexity is isolated as the equation $Y = xx'$. As others have done, this latter equation was relaxed to the convex semidefinite inequality $Y \succeq xx'$, which was then treated via a polyhedral outer approximation (to avoid a method that will be tied to the difficulty of solving many semidefinite programs). The contribution of [33,34] was to apply disjunctive-programming methodology to the nonconvex inequality $Y \preceq xx'$, by seeking one-dimensional quadratic concavity within this nonconvex inequality. Further disjunctive cuts are employed related to integer variables and linear complementarity constraints, when those were present in test instances. Although our present approach is quite different, we plan to absorb the lessons learned from [33,34], and we will apply disjunctive-cut techniques wherever we can efficiently exploit them. Moreover, [33,34] did not incorporate their bounding methodology into a full spatial branch-and-bound for global optimization. We see advantages in our approach with such a goal in mind. In particular, our methodology appears to be much less computationally expensive, and it should be more numerically stable.

Acknowledgements We gratefully acknowledge the Newton Institute for partial support. J. Lee was partially supported by NSF Grant CMMI–1160915. M. Fampa was partially supported by a Research Grant from CNPq-Brazil. W. Melo was visiting the University of Michigan, supported by a Research Fellowship from CNPq-Brazil. We are also grateful to Sam Burer for providing us the BoxQP instances.

References

1. Claire S. Adjiman, Stefan Dallwig, Christodoulos A. Floudas and Arnold Neumaier, A global optimization method, αBB, for general twice-differentiable constrained NLPs — I. Theoretical advances. Computers & Chemical Engineering, 22(9):1137–1158, 1998.
2. Claire S. Adjiman, Stefan Dallwig, Christodoulos A. Floudas and Arnold Neumaier, A global optimization method, αBB, for general twice-differentiable constrained NLPs — II. Implementation and computational results. Computers & Chemical Engineering, 22(9):1159–1179, 1998.
3. Farid Alizadeh, Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization, SIAM Journal on Optimization, v. 5, 13–51, 1993.
4. Kurt M. Anstreicher, Marcia Fampa, Jon Lee and Joy Williams, Continuous relaxations for constrained maximum-entropy sampling. In: "Integer Programming and Combinatorial Optimization (Vancouver, BC, 1996)", volume 1084 of Lecture Notes in Computer Science, pages 234–248. Springer, Berlin, 1996.
5. Kurt M. Anstreicher, Marcia Fampa, Jon Lee and Joy Williams, Using continuous nonlinear relaxations to solve constrained maximum-entropy sampling problems. Mathematical Programming, Series A, 85:221–240, 1999.
6. Kurt M. Anstreicher, Marcia Fampa, Jon Lee and Joy Williams, Maximum-entropy remote sampling. Discrete Applied Mathematics, 108:211–226, 2001.
7. Kurt Anstreicher and Jon Lee, A masked spectral bound for maximum-entropy sampling. In: A. Di Bucchianico, H. Läuter and H.P. Wynn, editors, "MODA 7 - Advances in Model-Oriented Design and Analysis", Contributions to Statistics, Springer, Berlin, 1–10, 2004.
8. Pietro Belotti, Jon Lee, Leo Liberti, François Margot, and Andreas Wächter, Branching and bounds tightening techniques for non-convex MINLP. Optimization Methods and Software, 24:597–634, 2009.

9. Pierre Bonami, Larry Biegler, Andrew Conn, Gérard Cornuéjols, Ignacio Grossmann, Carl Laird, Jon Lee, Andrea Lodi, François Margot, Nick Sawaya and Andreas Wächter, An algorithmic framework for convex mixed integer nonlinear programs. Discrete Optimization, 5:186–204, 2008.
10. Pierre Bonami, Jon Lee, Sven Leyffer and Andreas Wächter, On Branching Rules for Convex Mixed-Integer Nonlinear Optimization. To appear in : ACM Journal of Experimental Algorithmics.
11. S.P. Bradley, A.C. Hax and T.L. Magnanti, "Applied Mathematical Programming". Addison-Wesley, 1977.
12. Samuel Burer and Jon Lee, Solving maximum-entropy sampling problems using factored masks. Mathematical Programming, Volume 109, Numbers 2-3, 263–281, March, 2007.
13. Sam Burer and Anureet Saxena, The MILP road to MIQCP, pp. 373–405, In: Jon Lee and Sven Leyffer, Editors, "Mixed Integer Nonlinear Programming". The IMA Volumes in Mathematics and its Applications, Vol. 154, 2012.
14. Samuel Burer and Dieter Vandenbussche, A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. Mathematical Programming, Series A, 113:259–282, 2008.
15. Rainer E. Burkard, Eranda Çela, Panos M. Pardalos and Leonidas S. Pitsoulis, The quadratic assignment problem. Handbook of combinatorial optimization, Vol. 3, 241–237, Kluwer Acad. Publ., Boston, MA, 1998.
16. Paul H. Calamai, Luis N. Vicente and Joaquim J. Júdice, A new technique for generating quadratic programming test problems. Mathematical Programming 61: 215–231, 1993.
17. COUENNE, projects.coin-or.org/Couenne.
18. Claudia D'Ambrosio, Jon Lee and Andreas Wächter, An algorithmic framework for MINLP with separable non-convexity, pp. 315–347, In: Jon Lee and Sven Leyffer, Editors, "Mixed Integer Nonlinear Programming." The IMA Volumes in Mathematics and its Applications, Vol. 154, 2012.
19. Farhad Ghassemi and Vikram Krishnamurthy, A cooperative game-theoretic measurement allocation algorithm for localization in unattended ground sensor networks. 11th International Conference on Information Fusion, 1580–1586, 2008.
20. Roger Fletcher, Stable reduced hessian updates for indefinite quadratic programming. Mathematical programming, 87(2):251–264, 2000.
21. Roger Fletcher and Sven Leyffer, User manual for filterSQP, 1998. University of Dundee Numerical Analysis Report NA-181.
22. Raymond Hemmecke, Mathias Köppe, Jon Lee, Robert Weismantel, Nonlinear integer programming. In: M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi and L. Wolsey (eds.), "50 Years of Integer Programming 1958–2008: The Early Years and State-of-the-Art Surveys," Springer-Verlag, 2010, pp. 561–618.
23. Alan Hoffman, Jon Lee and Joy Williams, New upper bounds for maximum-entropy sampling. In: A.C. Atkinson, P. Hackl and W. G. Mller, editors, "MODA 6 - Advances in Model-Oriented Design and Analysis", Contributions to Statistics, Springer, Berlin, 143–153, 2001.
24. Chun-Wa Ko, Jon Lee and Maurice Queyranne, An exact algorithm for maximum entropy sampling. Operations Research, 43(4):684–691, 1995.
25. Jon Lee, Discussion on: 'A state-space-model approach to optimal spatial sampling design based on entropy'. Environmental and Ecological Statistics, 5:45–46, 1998.
26. Jon Lee, Constrained maximum-entropy sampling. Operations Research, 46:655–664, 1998.
27. Jon Lee, Semidefinite programming in experimental design. In: H. Wolkowicz, R. Saigal and L. Vandenberghe, editors, "Handbook of Semidefinite Programming", International Series in Operations Research and Management Science, Vol. 27, Kluwer, 2000.
28. Jon Lee, Maximum entropy sampling. In: A.H. El-Shaarawi and W.W. Piegorsch, editors, "Encyclopedia of Environmetrics". Wiley, 2001. Up-to-date version in: 2nd edition, in press.
29. Jon Lee, In situ column generation for a cutting-stock problem. Computers & Operations Research, Volume 34, Issue 8, August 2007, Pages 2345–2358.
30. Jon Lee and Sven Leyffer, Editors, "Mixed Integer Nonlinear Programming." The IMA Volumes in Mathematics and its Applications, Vol. 154, 2012.

31. Jon Lee and Joy Williams, A linear integer programming bound for maximum-entropy sampling. Mathematical Programming, Series B, 94:247–256, 2003.
32. Franz Rendl, Giovanni Rinaldi and Angelika Wiegele, Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Mathematical Programming 121 (2010), no. 2, Ser. A, 307–335.
33. Anureet Saxena, Pierre Bonami and Jon Lee, Convex relaxations of non-convex mixed integer quadratically constrained programs: Extended formulations, Mathematical Programming, Series B, 124(1–2):383–411, 2010.
34. Anureet Saxena, Pierre Bonami and Jon Lee, Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations, Mathematical Programming, Series A, 130(2):359–413, 2011.
35. Mohit Tawarmalani and Nikolaos V. Sahinidis, A polyhedral branch-and-cut approach to global optimization, Mathematical Programming, 103(2):225–249, 2005.
36. Hanbiao Wang, Kung Yao, Greg Pottie and Deborah Estrin, Entropy-based Sensor Selection Heuristic for Target Localization. In: Proceedings of the Third International Symposium on Information Processing in Sensor Networks, 36–45, ACM Press, 2004.